Appendix D

This Appendix describes a Macro Library for a board according to the present invention. The library contains functions for

1) Memory arbitration
2) Flash bus arbitration
3) Read and Write to Flash RAM
4) FPCOM settings
5) Control of the LEDs

```
////////////////////////////////////////////////////////////////
//
// Interfaces
//
// Shared RAM arbitration
//          ----------------------
//          KRequestMemoryBank(bankMask)
//          KReleaseMemoryBank(bankMask)
//
//          Flash RAM Macros
// ----------------
//          KEnableFlash()
//          KDisableFlash()
//          KSetFlashAddress(address)
//          KWriteFlashData(address, data)
//          KReadFlashData(address, data)
//          KReadFlashID(flash_component_ID, manufacturer_ID)
//
//
```

09725345 .012901

```
//      Flash bus arbitration
//      ---------------------
//      KSetFPGAFBM()
//      KReleaseFPGAFBM()
//
//      Others
//      ---------
//      KSetLEDs(maskByte)
// KSetFPCOM(fpcom)




#ifndef _KOMPRESSOR_LIBRARY
#define _KOMPRESSOR_LIBRARY

// Include header file
//#include "KompressorMaster.h"



/////////////////////////////////////////////////////////////////////////
// Request access to a memory bank
//
// The procedureS will block until access to all the requested banks have been
// granted.
//
```

09772534.012601

```
unsigned 1 shared_bank0_request = 1 with { warn = 0} ;
unsigned 1 shared_bank1_request = 1 with { warn = 0} ;

interface bus_out() shbk0req(shared_bank0_request) with
sram_shared_bank0_request_pin;
interface bus_out() shbk1req(shared_bank1_request) with
sram_shared_bank1_request_pin;
interface bus_clock_in(unsigned 1) shbk0grant() with sram_shared_bank0_grant_pin;
interface bus_clock_in(unsigned 1) shbk1grant() with sram_shared_bank1_grant_pin;



macro proc KRequestMemoryBank0()
{
        shared_bank0_request = 0;
        while(shbk0grant.in) delay;
}



macro proc KRequestMemoryBank1()
{
        shared_bank1_request = 0;
        while(shbk1grant.in) delay;
}
```

09725374 .012601

```
////////////////////////////////
// Release a memory bank
//


macro proc KReleaseMemoryBank0()
{
        shared_bank0_request = 1;
}


macro proc KReleaseMemoryBank1()
{
        shared_bank1_request = 1;
}





//////////////////////////////////////////////
//
// Functions for dealing with FP commands

#define FP_SET_IDLE             (unsigned 3)  7
#define FP_READ_STATUS (unsigned 3)  5
#define FP_CCLK_LOW             (unsigned 3)  3
```

09725344.012601

```
#define FP_CCLK_HIGH     (unsigned 3)   7
#define FP_WRITE_CONTROL (unsigned 3)       0


unsigned 3 fpcom = FP_SET_IDLE  with { warn = 0}; // default
interface bus_out() fpcom_bus(fpcom) with FPcom_pins;

macro proc KSetFPCOM(command)
{
        fpcom = command;
        delay;
        delay;
}


macro proc KReadCPLDStatus(status)
{
   par
        {
   KDisableFlash();
         flash_write = 0;
        }

   KSetFPCOM(FP_READ_STATUS);


         delay;
         delay;
         delay;
   delay;
```

T0620 "4E5W2760

```
    status = flash_data_bus.in;

    par
          {
                  KSetFPCOM(FP_SET_IDLE);
                  KEnableFlash();
          }
  }


  macro proc KWriteCPLDControl(control)
  {
          KDisableFlash();
          par
          {
                  flash_data  = (unsigned 8) (0 @ control);
                  flash_write = 1;
          }

          KSetFPCOM(FP_WRITE_CONTROL);
          delay;
          delay;
          delay;
          par
          {
                  KSetFPCOM(FP_SET_IDLE);
                  flash_write = 0;
                  KEnableFlash();
```

```
        }
}


//////////////////////////////////
//
//      Flash RAM stuff
//
//
// Parameters;
//
//      Read/write cycle            120ns
//      Address to output           120ns
//      CE to ouput                          120ns
//
//      CE low to WE low            0
//      write pulse width low  70ns
//      data setup to we high  50ns
//      address setup to we hi 55ns
//      address/data hold           0ns
//      write pulse width high30ns



unsigned 24 flash_address  with { warn = 0};
unsigned 8 flash_data  with { warn = 0};
unsigned 1 flash_cs = 1, flash_we = 1, flash_oe = 1  with { warn = 0}; // initialise to
high
```

10972534.012601

```
unsigned 1 flash_write = 0 with { warn = 0}; // controls direction of the data pins
unsigned 1 flash_on = 0 with { warn = 0};  // controls the other tristate buses

interface bus_ts_clock_in(unsigned 24)  flash_address_bus(flash_address, flash_on)
with {data = FA_pins};
interface bus_ts_clock_in(unsigned 1)  flash_chipselect(flash_cs, flash_on) with
flash_cs_pin;
interface bus_ts_clock_in(unsigned 1)  flash_writeenable(flash_we, flash_on) with
flash_we_pin;
interface bus_ts_clock_in(unsigned 1)  flash_outputenable(flash_oe, flash_on) with
flash_oe_pin;
interface bus_ts_clock_in(unsigned 8) flash_data_bus(flash_data, flash_write) with
{data = FD_pins};


macro proc KEnableFlash()
{
        par
        {
        flash_on = 1;
        flash_cs = 0;
        }
}


macro proc KDisableFlash()
{
        par{
        flash_on = 0;
```

10597254.012501

```
        flash_cs = 1;
        }
}


// Sets up the address on the
macro proc KSetFlashAddress(address)
{
        flash_address = address;
}


macro proc KWriteFlashData(address, data)
{

        par // set up address and data and drive onto pins
        {
        flash_oe = 1; // disable output
        flash_address = address;
        flash_data = data;
        flash_write = 1;
        flash_we = 0;  // send write pulse
        }

        // running at 50/2 MHz - 40 ns cycles - 2 delays should be
        // sufficient to meet timing constraint

        delay;
```

09725534.012901

```
  delay;

        par
        {
                flash_we = 1;
                flash_write = 1;
        }

}

macro proc KReadFlashData(address, data)
{
        par
        {
        flash_write = 0;
        flash_oe = 0; // enable output
        flash_address = address;
        }

        // running at 50/2 MHz - 40 ns cycles - 2 delays should be
        // sufficient to meet timing constraint
        delay;
   delay;
        data = flash_data_bus.in;

}


macro proc KReadFlashID(flashid, manid)
```

10972534.012601

```
{
        par
        {
                KEnableFlash();
                KSetFPGAFBM();
        }

        KWriteFlashData(0, 0x90);
        KReadFlashData(0, manid);
        KReadFlashData(2, flashid);

        par
        {
        KReleaseFPGAFBM();
        KDisableFlash();
        }

}

macro proc KReadFlashStatus(status)
{
                par
                {
                        KEnableFlash();
                        KSetFPGAFBM();
                }
```

09647254.072501

```
        KWriteFlashData(0, 0x70);
        KReadFlashData(0, status);

        par
        {
                KDisableFlash();
                KReleaseFPGAFBM();
        }

}



////////////////////////////////////
// Flash bus arbitration pins
//
unsigned 1 fbus_master = 1  with {warn = 0}; // initialise to not master
interface bus_out() bus_master_line(fbus_master) with BUSMaster_pin;

macro proc KSetFPGAFBM()
{
        fbus_master = 0;
}



macro proc KReleaseFPGAFBM()
{
        fbus_master = 1;
}
```

09723594.012501

```
/////////////////////////////////////////////////////
// LED control macros


unsigned 8 LED = 0  with {warn = 0}; // by default
unsigned 1 LED_en = 0 with {warn = 0};
interface bus_ts(unsigned 8) LEDpins(LED, LED_en) with LED_pins;
macro proc KSetLEDs(maskByte)
{
   par
   {
         LED = maskByte;
   LED_en = 1;
   }
}



//////////////////////////////////
//
// FPcom ==7 CCLK = High
//
// From the FPGA BUSMuster pin should be brought low and the FLASH may be
// accessed as any normal device RAM device.
//
#endif _KOMPRESSOR_LIBRARY
```

09725344.012501